Parallel Algorithm for Spherical Delaunay Triangulations and Spherical Centroidal Voronoi Tessellations

Douglas W. Jacobsen[a,*], Max Gunzburger[a], Todd Ringler[b], John Burkardt[a], Janet Peterson[a]

[a] *Department of Scientific Computing, Florida State University, Tallahassee, FL 32306, USA*
[b] *Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA*

Abstract

Spherical centroidal Voronoi tessellations (SCVT) are used in many applications in a variety of fields, one being climate modeling. They are a natural choice for spatial discretizations on the Earth, or any spherical surface. The climate modeling community, which has started to make use of SCVTs, is beginning to focus on exa-scale computing for large scale climate simulations. As the data size increases, the efficiency of the grid generator becomes extremely important. Current high resolution simulations on the earth call for a spatial resolution of about 15km. In terms of an SCVT this corresponds to a quasi-uniform SCVT with roughly 2 million Voronoi cells. Computing this grid serially is very expensive and can take on the order of weeks to converge sufficiently for the needs of climate modelers. This paper outlines a new algorithm that utilizes existing computational geometry tools such as conformal mapping techniques, planar triangulation algorithms, and basic domain decomposition, to compute SCVTs in parallel, thus reducing the overall time to convergence. This new algorithm shows speedup on the order of 4000 when using 42 processors over STRIPACK in computing a triangulation used for generating an SCVT.

keywords: Spherical Centroidal Voronoi Tessellation, Spherical Delaunay Triangula-

*Corresponding Author E-mail Address: dwj07@fsu.edu

tion, Parallel, Stereographic Projection, Domain Decomposition, High Performance Computing

1. Introduction

Over the past few decades Voronoi diagrams have become a natural choice for spatial discretizations due to their ability to handle arbitrary boundaries, densities, and refinement well. These grids can be used in a wide range of applications, and can be created for almost any geometry, including multi-dimensional space. However, creating these meshes can be overly time consuming, especially for high quality, high resolution grids. To attempt to speed up grid generation, several groups have worked on creating parallel divide-and-conquer algorithms to handle the construction of Delaunay triangulations ([1],[2]), which are the dual grids of Voronoi diagrams. The few algorithms that do exist for the parallel construction of Delaunay triangulations are limited to two-dimensional planar surfaces.One specific field which has recently begun adopting Voronoi diagrams, in addition to other unstructured meshes for spatial discretizations, is climate modeling ([3], [4]). With a recent trend towards exa-scale in most aspects of high performance computing, there is a demand for fast algorithms to generate high resolution spatial meshes. Also of interest are variable resolution grids with smooth transition regions [5].

With the current need for a high resolution spherical grid generator in mind, we combine tools in computational geometry to allow for fast generation of spherical centroidal Voronoi tessellations. These tools lead to a new algorithm which makes use of stereographic projections, and a novel method for domain decomposition. This paper outlines this new algorithm, designed to allow for the parallel computation of spherical centroidal Voronoi tessellations that may tessellate the entire sphere, or some subregion of interest. The paper is organized in the following fashion: Section 2 will cover some background material as well as detail the new algorithm, Section 3 will present some results on actual grids generated and performance results for the parallel algorithm, and Section 4 will

conclude with a discussion.

2. Theory/Calculation

2.1. Delaunay Triangulations

A k-simplex is defined as a set of $k+1$ points that are the vertices of the k-simplex. For example, a 2-simplex would be a triangle, and a 3-simplex would be a tetrahedron. A k-simplex is made up of what are referred to as s-faces, where a s-face is made up of any $s+1$ distinct vertices of the k-simplex. For example, a 2-face is a triangular face, a 1-face is an edge, and a 0-face is a vertex.

Given a point set, P, in $\mathbb{R}^d$, the Delaunay triangulation of this point set, $D(P)$, is the set of d-simplices such that:

- A point, p, in $\mathbb{R}^d$, is a vertex of a simplex in $D(P)$, $\iff$ $p \in P$;
- The intersection of two simplices in $D(P)$, is either the empty set, or a common face;
- The interior of the circumscribing d-sphere through the $k+1$ vertices of a particular simplex contains no other points from the set P.

If the circumscribing d-sphere has more than $k+1$ points lying on its perimeter, the triangulation is Delaunay, but not unique. The Delaunay triangulation of a point set defined in $\mathbb{R}^d$ is related to the convex hull of the point set when projected onto a paraboloid in $\mathbb{R}^{d+1}$ [1].

2.2. Voronoi Tessellations

The dual mesh of a Delaunay triangulation is called the Voronoi tessellation. Given a set of points, P, called generators, the Voronoi tessellation, $V = V_i$, is defined as

$$||\mathbf{x} - \mathbf{x}_i|| < ||\mathbf{x} - \mathbf{x}_j|| \ \forall \mathbf{x} \in V_i, \tag{1}$$

where V_i represents a Voronoi cell, and $\mathbf{x}_i \in P$ and $\mathbf{x}_j \in P$ represent generators. This property, called the Voronoi property, states that every point contained

inside a Voronoi cell is closer to its cell generator than to any other generator in the set P. To be a centroidal Voronoi tessellation, the cell generators $\mathbf{x}_i$ are required to be the centers of mass for the cells, meaning $\mathbf{x}_i = \mathbf{x}_i^*$, with $\mathbf{x}_i^*$ defined as

$$\mathbf{x}_i^* = \frac{\int_{V_i} \mathbf{x}\rho(\mathbf{x})dx}{\int_{V_i} \rho(\mathbf{x})dx}, \tag{2}$$

where $\rho(\mathbf{x})$ defines a non-negative point-density function which can be used to create variable resolution meshes.

The center of mass and the generator of a Voronoi cell are generally not coincident. The requirement that $\mathbf{x}_i$ and $\mathbf{x}_i^*$ be the same can be imposed through one of many algorithms, such as Lloyd's algorithm [6]. Lloyd's algorithm imposes this by iterating on the point set, moving each generator to its Voronoi cell's center of mass until they are identical. Lloyd's algorithm is more rigorously discussed in [7].

In general, the density function in (2) affects the grid spacing of the final SCVT. If we arbitrarily select two Voronoi cells from a tessellation, and index them i and j, their grid spacing and density are related as

$$\frac{h_i}{h_j} \approx \left[\frac{\rho(\mathbf{x}_j)}{\rho(\mathbf{x}_i)}\right]^{\frac{1}{d'+2}}, \tag{3}$$

where d' is the dimension of the simplical elements in the tessellation, $\rho(\mathbf{x}_i)$ is the density function as in (2) evaluated at a point $\mathbf{x}_i \in V_i$, and h_i is a measure of the local grid spacing at the point $\mathbf{x}_i$. Though (3) is an open conjecture, it has been supported through many numerical studies as can be seen further in [5].

Replacing all of the constructs defined in Sections 2.1 and 2.2 with their analogous components on the surface of a sphere creates the spherical complements to Delaunay triangulations and Voronoi tessellations. The spherical versions of Delaunay triangulations and Voronoi tessellations are used for the construction of SCVTs as opposed to planar CVTs which have been discussed above for simplicity. While planar CVTs tessellate a 2-dimensional region with polygons, an

SCVT tessellates the surface of a 3-dimensional sphere with polygons.

2.3. Stereographic Projections

Stereographic projections are special mappings between the surface of a sphere and a plane tangent to the sphere. Not only are stereographic projections a conformal mapping, meaning that angles are preserved, but the projections also preserve circularity. As will be discussed below, preserving circularity is a particularly important property of stereographic projections. Stereographic projections also map the interior of these circles to the interior of the mapped circles ([8], [9]). Preserving circularity implies that the stereographic projection preserves Delaunay criteria as shown in Section 2.1, because Delaunay triangle circumcircles (along with their interiors) are preserved, and therefore Delaunay triangulations are preserved. This projection can be used to compute a triangulation of a portion of the sphere, by allowing the triangulation to be carried out in the more convenient geometry of the plane.

To define the stereographic projection, we need to define the following quantities, all in Cartesian coordinates in $\mathbb{R}^3$. $\mathbf{C}$ is the center of the sphere, typically the origin, $\mathbf{T}$ is the point of tangency (where the projection plane is tangent to the sphere), $\mathbf{F}$ is the focus point, which is a reflection about $\mathbf{C}$ of $\mathbf{T}$, and $\mathbf{P}$ is a point on the surface of the sphere. The stereographic projection of $\mathbf{P}$ into a point $\mathbf{Q}$ on the plane is defined in (4) and (5).

$$s = 2 * \frac{(\mathbf{C} - \mathbf{F}) \cdot (\mathbf{C} - \mathbf{F})}{(\mathbf{C} - \mathbf{F}) \cdot (\mathbf{P} - \mathbf{F})} \tag{4}$$

$$\mathbf{Q} = s * \mathbf{P} + (1 - s) * \mathbf{F} \tag{5}$$

Figure 1 illustrates the stereographic projection, using the variables defined for (4) and (5).

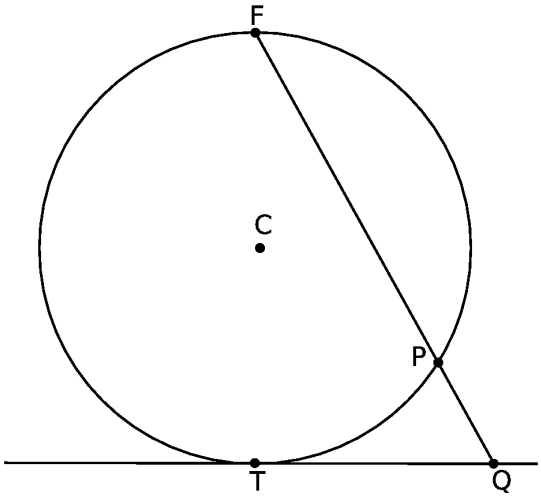


Figure 1: Cross-sectional illustration of a stereographic projection from a sphere into a tangent plane.

For the purposes of this paper, it is more useful to define the projection relative to $\mathbf{T}$, rather than $\mathbf{F}$. A simple substitution of $\mathbf{T} = \mathbf{C} - \mathbf{F}$ produces (6) and (7).

$$s = 2 * \frac{1}{(\mathbf{T}) \cdot (\mathbf{P} + \mathbf{T})} \tag{6}$$

$$\mathbf{Q} = s * \mathbf{P} + (s - 1) * \mathbf{T} \tag{7}$$

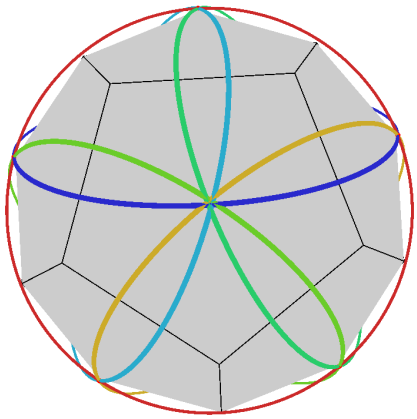
This projection can be used to project from $\mathbb{R}^d$ to $\mathbb{R}^{d-1}$, and can be repeated until $d - 1 = 2$.

2.4. Algorithm Details

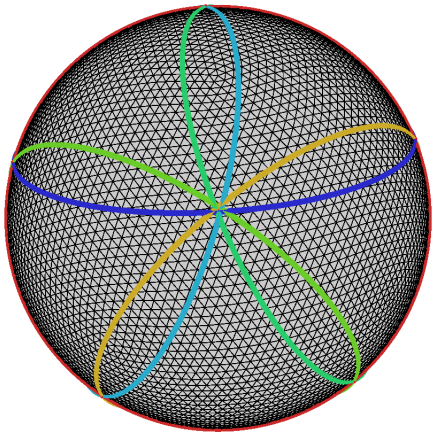
The parallel algorithm closely follows the layout of Lloyd's algorithm, with a few modifications. The key modification is computing a Delaunay triangulation in parallel, since all other portions are considered embarrassingly parallel. The idea of computing a planar triangulation in parallel has been discussed for several years [1]. Typically, such algorithms divide the point set up into smaller regions that can then be triangulated independently from each other. Each triangulation needs to be stitched together to form a global triangulation.

This stitching, or merge step, is typically computed in serial because it could involve modifying significant portions of each triangulation if the division was not performed correctly. The merge step is the main difference between most parallel algorithms. The main benefit of the presented algorithm is that the merge step is done in parallel. To create a spherical triangulation in parallel, a similar technique is employed as in the planar triangulations.

First, the sphere is divided into N overlapping regions $Y_k(\mathbf{T}_k, R_k)$ for $k = 1, \ldots, N$, which are defined by a geodesic arc R_k, and a tangent plane defined at the region's point of tangency $\mathbf{T}_k$. Each of these regions is owned by an independent processor, and these regions also have some connectivity, or list of neighbors, defined. On the sphere, these regions would look like overlapping umbrellas, as can be seen in figure 2(a). Each region (or processor) would take from the global point set, $\mathbf{p}_i \in P$, the points that are inside of its region radius, where $\cos^{-1}(\mathbf{T}_k \cdot \mathbf{p}_i) \leq R_k$. Keep in mind, this sorting may cause one point may be in several regions, as in figure 2, where figure 2(a) shows an example domain decomposition with 12 regions that could be used on a set of generators shown triangulated in figure 2(b). Since the end goal of this algorithm is to compute an SCVT, the regional triangulations do not need to be merged on every iteration because they overlap.



(a) 12 Generator SCVT



(b) 10242 Generator Delaunay Triangulation

Figure 2: Domain Decomposition Example. Figure 2(a) is an SCVT used for a 12 processor domain decomposition, where figure 2(b) is an 10242 generator Delaunay triangulation computed using the 12 generator SCVT for parallelization. Each colored ring represents a regions radius R_k, where region centers $\mathbf{C}_k$ are the Voronoi cell center, at the center of each pentagonal structure in 2(a).

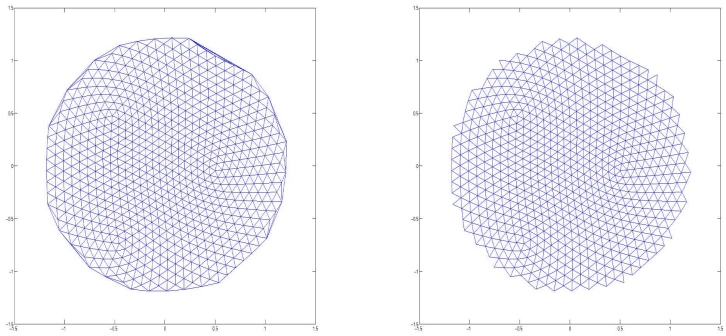
$\tilde{\mathbf{P}}_k = S[\hat{\mathbf{P}}_k, \mathbf{T}_k]$. After a spherical point set $\hat{\mathbf{P}}_k$ is determined, $\tilde{\mathbf{P}}_k = S[\hat{\mathbf{P}}_k, \mathbf{T}_k]$ where $S[\hat{\mathbf{P}}_k, \mathbf{T}_k]$ represents the stereographic projection of $\hat{\mathbf{P}}_k$ into the plane tangent to the sphere at point $\mathbf{T}_k$. Because a stereographic projection preserves circles (and their interiors), the projection also preserves the Delaunay

criteria that every triangle's circumcircle needs to be empty. The newly projected point set is now triangulated using some planar triangulation algorithm, such as Triangle [10] which is used in this study. If the mapping from global point index to local point index is appropriately maintained, a simple map from local index to global index gives the approximate triangulation for the region on the sphere. One final step is needed to make this the true triangulation for the region, which is to remove all "non-Delaunay" triangles. The criteria required to be a Delaunay triangle in the global triangulation is defined in (8) as

$$\cos^{-1} ||\mathbf{T}_k - \hat{\mathbf{c}}_i|| + \hat{r}_i < R_k, \tag{8}$$

where $\mathbf{T}_k$ is a region center, R_k is a region radius, $\hat{r_i}$ is a triangle circumradius, and $\hat{\mathbf{c}}_i$ is a triangle circumcenter.

Since each region is unaware of the triangles and points outside of its radius, only triangles whose circumcircles are completely contained inside of the region radius R_k are guaranteed to be Delaunay, as no other points from the point set can be in their circumcircle. Any triangle whose circumcircle extends outside of its regions radius may contain points that were not in the $\hat{P}_k$, and should be discarded from the region's triangulation because this triangle may not be adhere to the Delaunay criteria. Figure 2.4 visualizes this point, where 3(a) shows a projected planar triangulation $\tilde{P}_k$ before removing triangles that do not satisfy (8), and 3(a) shows the exact same triangulation after removing these potentially non-Delaunay triangles. After this step is complete, the regional triangulation is now exactly Delaunay. After the regional triangulation is computed, the integration step of Lloyd's algorithm can begin. The overlapping of regions is key to this portion of the algorithm, because if the overlap of regions is not large enough some true Delaunay triangles may not lie entirely in at least one region.



(a) Before application of (8)

(b) After application of (8)

Figure 3: Triangulations in a plane after Stereographic projection. 3(a) is the triangulation before (8) is applied, and 3(b) is after it is applied

In Lloyd's algorithm, after the Delaunay triangulation of the point set is computed, every Voronoi cell center of mass must be computed by integration, so its generator can be replaced. This step typically requires the computation of the Voronoi diagram for a region in addition to the Delaunay triangulation previously computed. However, some careful geometry can reveal that one doesn't actually need the Voronoi diagram. A single triangle from a Delaunay triangulation contributes to the integration of three different Voronoi cells. As seen in figure 4, if the triangle is split into three kites, each made up of two edge midpoints, the triangles circumcenter, and a vertex of the triangle, each one contributes to the Voronoi cell associated with the triangle vertex that is part of the kite. Integrating each kite, and updating a portion of the centroid integral allows one to only use the Delaunay triangulation when computing a CVT or an SCVT, so that no mesh connectivity needs to be computed on an iteration basis.

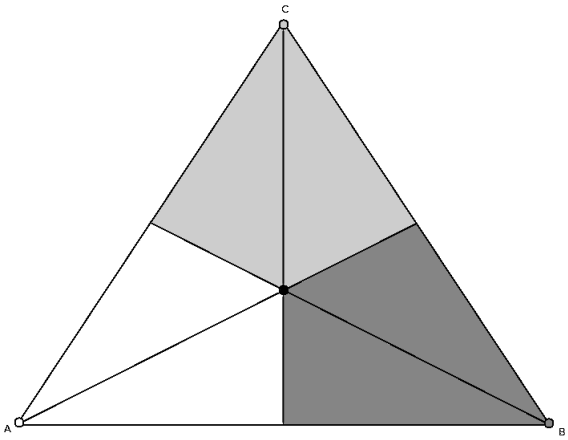


Figure 4: Triangle division used for integrating Voronoi cells using only the Delaunay triangulation without any adjacency information. Kite sections contribute to the Voronoi cell centered at the vertex that is part of the kite. A, B, C vertices are generators in the point set, where the point at the center of the triangle is the circumcenter of this triangle. Triangular regions that are colored similarly contribute to the same vertex.

To make this algorithm parallel, one simply has to ensure that each generator is only updated by one region. This can be done using one of a variety of domain decomposition methods. The method used in this particular algorithm uses the set of generators from a coarse SCVT to define region centers. Each region then updates only the generators that are inside of its defined Voronoi cell based on (1), using region centers $\mathbf{T}_k$ as $\mathbf{x}_i$ and generators $\mathbf{p}_i \in P$ as $\mathbf{x}$. Since Voronoi cells are non-overlapping, each generator will only get updated by one region. As mentioned earlier, the overlapping of regions is necessary to ensure that the triangulation of all points contained inside each region's Voronoi cells is exact. In practice, a region radius corresponding to the maximum distance to any adjacent region center allows enough overlap for the triangulation to be exact define in (9) as

$$R_i = \max_{j=1,\ldots,N} \cos^{-1}(\mathbf{T}_i \cdot \mathbf{T}_j), \tag{9}$$

where N is the number of region neighbors, $\mathbf{T}_i$ is the region center of interest, $\mathbf{T}_j$ is a neighboring region center, and R_i is the geodesic arc distance for region i.

While this heuristic allows the algorithm to work correctly, it may not be optimal for variable resolution grids, as some regions might contain many more points than they need to when they border both a fine and a coarse region.

Once each of the generators is updated, each region needs to transfer its newly updated points only to its adjacent neighbors, not to all of the active processors. This limits each processors communications to roughly 6 sends and receives, regardless of the total number of processors used. After this step is over, the convergence of the grid is checked, and the iterations continue, or stop depending on the result.

2.4.1. Convergence Criteria

When checking for convergence, two metrics are used. Currently, the L_2 norm (10) of the generator movement and the L_∞ norm (11) of generator movement are compared with some tolerance. If either norm reaches tolerance, the iteration process is deemed to have converged. The L_∞ is more strict, but both of these norms follow similar convergence paths when plotted against iteration number. There are other grid metrics that can be used, such as the clustering energy [11] as in (12), but in practice this tends to be less strict, and more computationally expensive, when compared with generator movement.

$$L_2 = \frac{\sqrt{\sum_{i=1}^{N_{pts}} (\mathbf{x}_i^n - \mathbf{x}_i^{n+1})^2}}{N_{pts}} \tag{10}$$

$$L_\infty = \max_{i=1,\ldots,N_{pts}} (|\mathbf{x}_i^n - \mathbf{x}_i^{n+1}|) \tag{11}$$

$$CE = \sum_{i=1}^{N_{pts}} \int_{V_i} (\rho(\mathbf{x})||\mathbf{x} - \mathbf{x}_i||^2 dx) \tag{12}$$

2.4.2. Initial Conditions

A variety of initial conditions can be used in an SCVT generator. The most obvious is Monte Carlo points [12]. These can either be uniformly distributed over the sphere, to create a quasi-uniform initial condition, or they can be sampled using the target density function, to potentially reduce the number of

iterations required for convergence. In addition to using Monte Carlo initial conditions, one can use a bisection method to build fine grids from a coarse grid [13]. To create a bisection grid, a coarse grid will be converged, using as few points as possible. After this coarse grid is converged, the midpoint of every Voronoi cell edge, or Delaunay triangle edge is added to the set of points. This causes the overall grid spacing to be reduced by roughly a factor of two in every cell. It also makes the point set roughly four times as large. In this paper, uniform Monte Carlo points and the bisection method will be used to compare timings for grid creation, however there are many other options one could use as an initial condition.

3. Results

Two different types of grids are presented to show the robustness of this algorithm. To begin, quasi-uniform meshes are created, followed by more complicated variable resolution meshes that cover the entire sphere. This method can also be used to create limited area grids on the sphere, however this will not be discussed in this paper.

For all of these results, serial versions were computed on an Intel Core 2 Duo T8100 CPU with 3GB of RAM, and parallel versions were computed using Florida State University's High Performance Computing Facility.

3.1. Quasi-Uniform results

STRIPACK [14] is an ACM TOMS algorithm that computes Delaunay triangulations on a sphere. STRIPACK is a serial code used as a baseline for comparison in this study. It is currently one of the few well-known spherical triangulation libraries available, and is written in Fortran 77. Figure 5 shows the performance of STRIPACK [14] as the number of generators is increased through bisection as mentioned in Section 2.4.2. The green dashed line represent the portion of the code that performs the integration of the Voronoi cells and the red solid line represent the portion of the code that performs the Delaunay triangulation. It is clear that the majority of the time per iteration is spent

in computing the Delaunay triangulation, and as the number of generators increases the time spent computing a Delaunay triangulation grows more rapidly than the time to integrate all Voronoi cells.

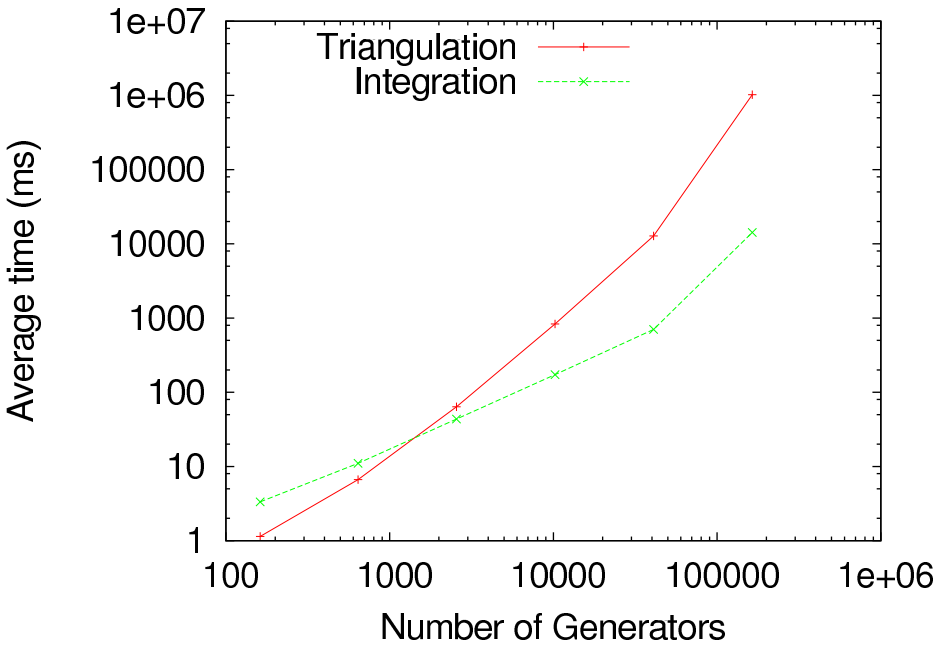


Figure 5: Timings for a STRIPACK based SCVT Generator at 162, 642, 10242, and 40962 generators. Red solid lines represent the time spent in STRIPACK computing a triangulation, where green dashed lines represent the time spent integrating the Voronoi cells outside of STRIPACK in one iteration of Lloyd's algorithm.

Since most climate models are shifting towards global high resolution simulations, the target quasi-uniform grid for this paper is a global 15km resolution grid, which corresponds to 2621442 grid points, or Voronoi cells. Grids created based on uniform Monte Carlo, and bisection initial conditions are compared. The time for these grids to converge to 10^{-6} in the L_2 norm, as in (10), is presented. A threshold of 10^{-6} is the strictest convergence levels that the Monte Carlo grid can attain, and is therefore chosen as the convergence threshold for this study. However, the bisection grid can converge well beyond this point. Table 1 shows timing results for the parallel algorithm comparing these two different options of initial conditions. It is clear from this table that bisection

initial conditions provide a significant speedup in the overall cost to generate a grid, seeing as it takes roughly 1/20th of the time to converge a bisection grid when compared to a Monte Carlo grid. Based on the results presented in table 1, only bisection initial conditions are used for the following experiments, unless otherwise specified.

Timed Portion	Bisection (B)	Monte Carlo (MC)	Speedup $\frac{MC}{B}$
Total Time (ms)	3,526,041	70,581,300	20.01
Triangulation Time (ms)	73,684	21,164,512	287.23
Integration Time (ms)	235,016	12,211,376	51.95
Communication Time (ms)	3,152,376	33,713,473	10.69

Table 1: Timing results for MPI-SCVT with Bisection and Monte Carlo initial conditions and the speedup of Bisection relative to Monte Carlo initial conditions

Tables 2 and 3 compare the algorithm described in this paper (MPI-SCVT) with STRIPACK [14], for computing spherical Delaunay triangulations. The results in these tables compare the cost to compute a single triangulation of a 163842 generator (60km global) grid. Table 2 compares STRIPACK with the final triangulation routine in MPI-SCVT. This routine produces a full triangulation of the entire sphere, and is only called once, at the very end of the grid generation process. The final triangulation routine involves each region computing its respective Delaunay triangulation. Due to the overlap in regions, the final triangulation is simply the unique union of all of these triangulations, keeping only one copy of each triangle.

Algorithm	Procs	Regions	Time (ms)	Speedup
STRIPACK	1	1	207528.81	Baseline
MPI-SCVT	1	2	9504.02	21
MPI-SCVT	42	42	5663.30	37

Table 2: Comparison of STRIPACK with Serial and Parallel versions of MPI-SCVT using final triangulations

Table 3 compares STRIPACK with the triangulation routine in MPI-SCVT that is called on every iteration. The results presented relative to MPI-SCVT in table 3 are averages over 2000 iterations. It is clear from this table that we see a

significant speedup over both the serial versions of MPI-SCVT and STRIPACK when using only 42 processors.

Algorithm	Procs	Regions	Time (ms)	Speedup
STRIPACK	1	1	207528.81	Baseline
MPI-SCVT	1	2	3623.09	57
MPI-SCVT	42	42	50.6572	4092

Table 3: Comparison of STRIPACK with Serial and Parallel versions of MPI-SCVT using per iteration triangulations

As was previously mentioned, the drastic different between Tables 2 and 3 is due to the different algorithms for computing triangulations. While Table 2 presents timings that are directly comparable to STRIPACK, Table 3 presents timings more useful in computing SCVTs.

As a comparison with figures 5, figures 6 and 7 present timing graphs made from MPI-SCVT. From these four plots, it is clear that the increase in time to compute the Delaunay triangulation does not grow as fast with problem size as it did in STRIPACK. Two processors are used, because this is the minimum amount of parallelization that MPI-SCVT supports, and MPI-SCVT requires at least 2 regions because the stereographic projection has a singularity at the focus point. Eventually, at around 163842 generators, the triangulation becomes more expensive than the integration step at least for 2 processors. As the number of processors increases this problem size might increase as well. Figure 6 represents the timings of MPI-SCVT for 2 regions as the problem size increases. Figure 7(a) represents the timings for a 40962 generator grid, which is a global 120km resolution, where figure 7(b) represents a 163842 generator grid, with a 60km resolution, and figure 7(c) represents a 2621442 generator grid with a 15km resolution.

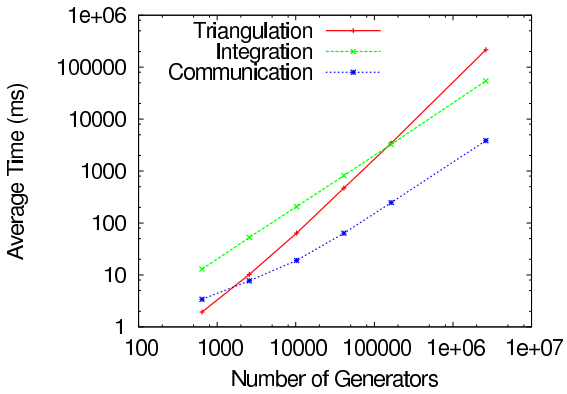
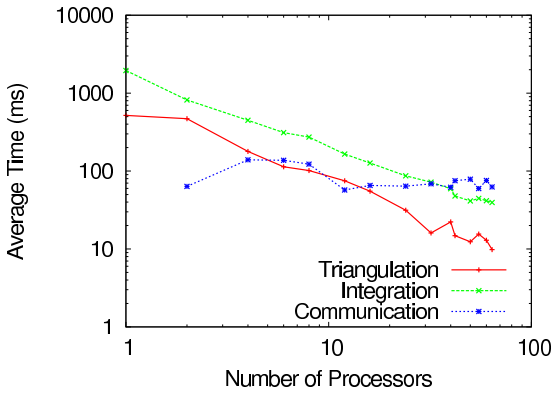
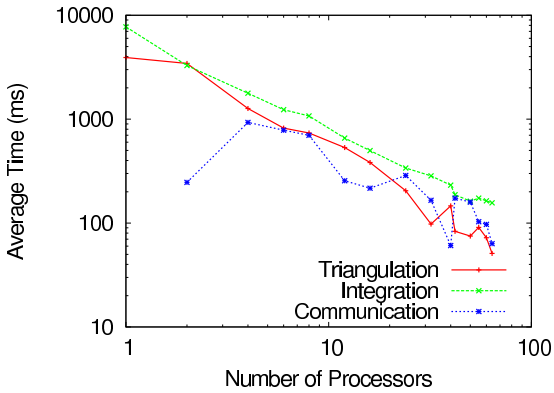


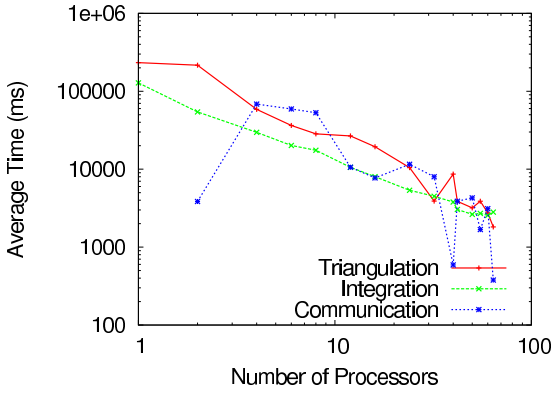
Figure 6: Timings for various portions of MPI-SCVT using 2 processors and 2 regions. As the problem size increases the slope of both the triangulation (Red-Solid) and the Integration (Green-Dashed) remain constant. The triangulation does not become more expensive than the integration until after roughly 163842 generators.



(a) 40962 Generator Timings



(b) 163842 Generator Timings



(c) 2621442 Generator Timings

Figure 7: Timing Results from MPI-SCVT vs. Number of processors. Constant problem size, shown as parallelization is increased. Red solid lines represent the cost of computing a triangulation, where green dashed lines represent the cost of integrating all Voronoi cells, and blue dotted lines represent the cost of communicating each regions updated point set to it's neighbors.

3.2. Variable Resolution results

Variable resolution grids here are only computed using MPI-SCVT. This is done because STRIPACK performs comparably in both the uniform case and variable resolution cases. The main issue in terms of variable resolution grids comes in to the domain decomposition used for MPI-SCVT. For example, a poor choice in domain decomposition could force the overlap in regions to be significantly larger than it needs to be. The larger the overlap of regions, the more points each region has to, needlessly, triangulate. This is especially apparent when using variable resolution grids as will be seen later. Because of this, two simple domain decompositions are used on a grid with a highly varying density function applied, in addition to one more complicated domain decomposition method. Timings are presented to determine which performs better, and gives better load balancing. The density function used to compute the grids in this Section can be seen in figure 8. The analytic form of the density function used is shown in figure 8 and expressed as (13).

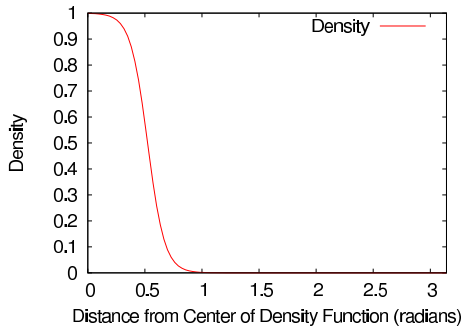
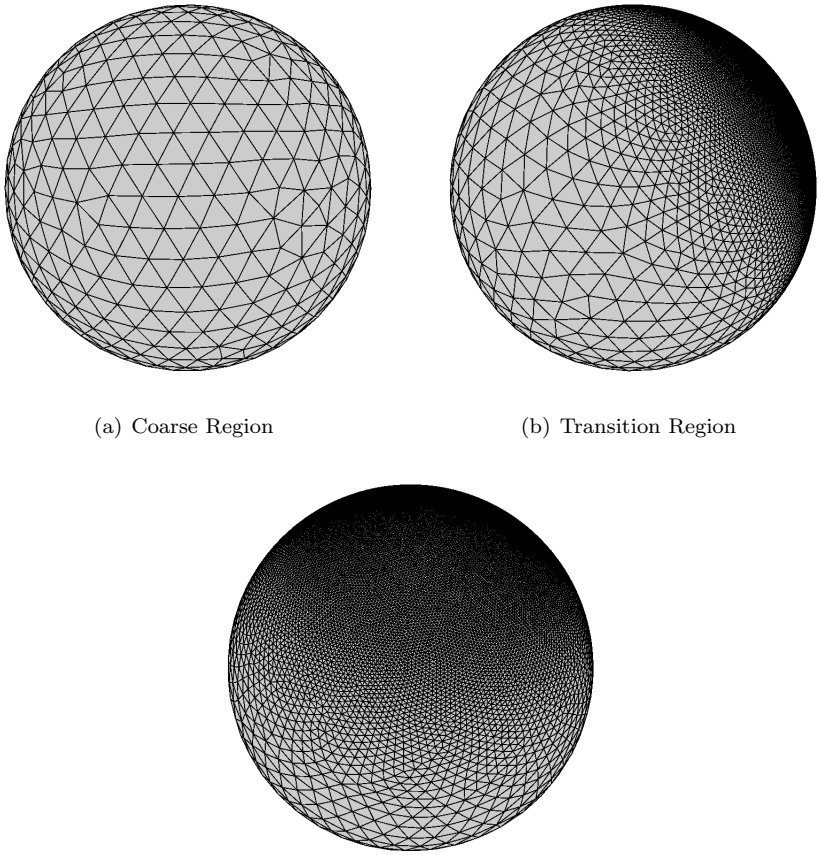


Figure 8: Density function that creates a grid with resolutions that differ by a factor of 16 between the coarse and the fine region. The maximum value of the density function is 1, where the minimum value is $(1/16)^4$.

$$\rho(\mathbf{x}_i) = \frac{1}{2\,(1-\gamma)}\left[\tanh\left(\frac{\beta - |\mathbf{x}_c - \mathbf{x}_i|}{\alpha}\right) + 1\right] + \gamma, \tag{13}$$

where $\mathbf{x_i}$ is constrained to lie on the surface of the unit sphere. This function results in relatively large value of ρ within a distance β of the point $\mathbf{x_c}$ where β is measured in radians and $\mathbf{x_c}$ is also constrained to lie on the surface of the sphere.

The function transitions to relatively small values of ρ across a radian distance of α. The distance between $\mathbf{x_c}$ and $\mathbf{x_i}$ is computed as $|\mathbf{x_c} - \mathbf{x_i}| = \cos^{-1}(\mathbf{x_c} \cdot \mathbf{x_i})$ with a range from 0 to π. Figure 9 shows an example grid created using this density function, with $\mathbf{x_c}$ set to be $\phi_c = 3\pi/2$, $\lambda_c = \pi/6$ where ϕ represents longitude and λ represents latitude, $\gamma = (1/16)^4$, $\beta = \pi/6$, and $\alpha = 0.15$ with 10242 generators. This set of parameters used in (13) is referred to as x16.



(a) Coarse Region

(b) Transition Region

(c) Fine Region

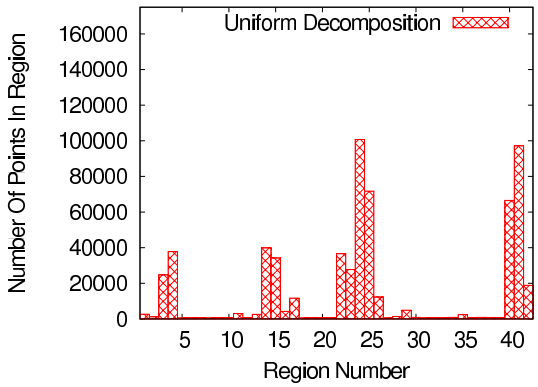
Figure 9: Figures show a variable resolution grid created using a density function with the format defined in (13). All three figures are of the same grid, only the viewing perspective is changed. Figure 9(a) shows the coarse region of the grid, 9(b) shows the transition region of the grid, and 9(c) shows the fine region of the grid.

It was previously mentioned in Section 2.4 that the heuristic used to deter-

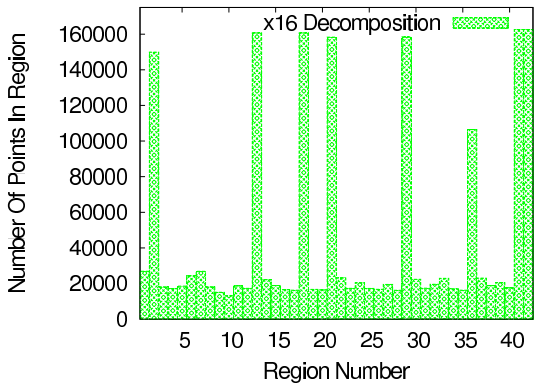
mine the region radius does not provide good load balancing with respect to variable resolution grids. To resolve this issue, a new algorithm for determining a regions point set is developed. The new algorithm begins by sorting points into all of the region's respective Voronoi cells. After all regions have their points in their Voronoi cells, the union of this point set with the neighboring Voronoi cell's point sets gives the final point set used. This sort method is more expensive to perform, however the better load balancing reduces idle computing time from processors that have small loads. Timings using this new method in addition to two dot-product-based methods for domain decomposition methods can be seen in table 4. Figure 10 shows the number of points that each processor has to triangulate on a per iteration basis. These timings and figures were computed using the exact same initial conditions, which was a converged x16 grid with 163842 generators, and they all used 42 processors, and 42 regions. Timings presented in table 4 are averages over 3000 iterations. Based on table 4 and figure 10 there is a significant advantage to the Voronoi based decomposition in that it not only speeds up the overall cost per iteration, but it provides a more balanced load across the processors. In table 4 note that the timings are taken relative to processor number 0, and as can be seen in figure 10(a), processor 0 has a very small load so the majority of its iteration time is spent waiting for the processors with large loads to finish and catch up which is included in the Communication column of the table.

Decomposition	Triangulation	Integration	Communication	Iteration	Speedup
Uniform	14.9779	39.3149	2556.971	2611.35	Base
x16	104.793	276.681	1560.71	1965.56	1.32
Voronoi	98.5482	249.77	288.694	640.472	4.07

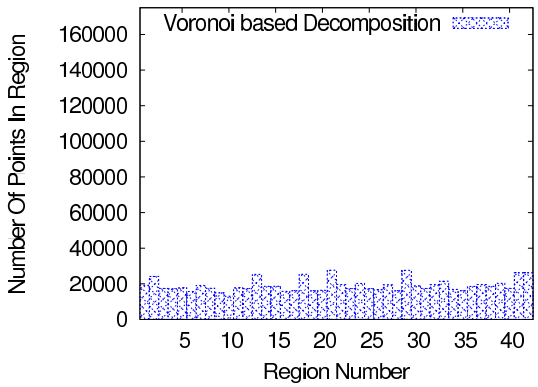
Table 4: Timings based on the domain decomposition used. Uniform uses a coarse Quasi-uniform SCVT to define region centers and their associated radii, and sorts using a simple dot product. x16 uses a coarse x16 SCVT to define region centers and their associated radii, and sorts using a simple dot product. Voronoi uses a coarse x16 SCVT to define region centers and their associated radii, and sorts using a Voronoi cell based sort.



(a) Uniform



(b) x16

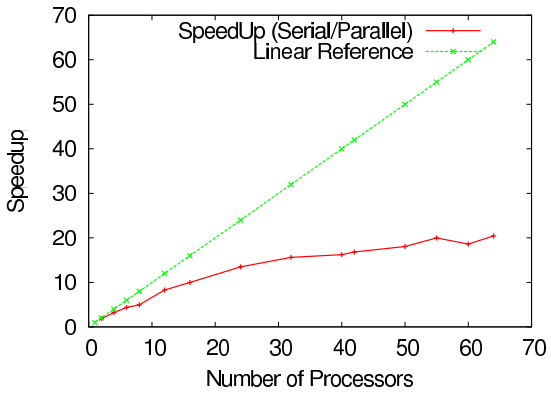


(c) Voronoi

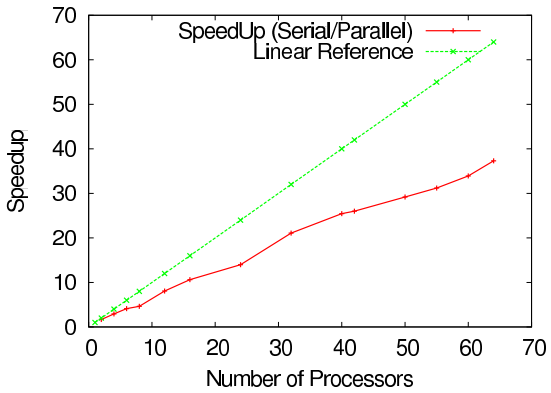
Figure 10: Number of points each processor has to triangulate. 10(a) uses a quasi-uniform SCVT for the decomposition, with a simple dot product. 10(b) uses a x16 SCVT for the decomposition, with a simple dot product. 10(c) uses a x16 SCVT for the decomposition, with a more complicated sort based on the region's Voronoi diagram.

3.3. Grid Generator Performance

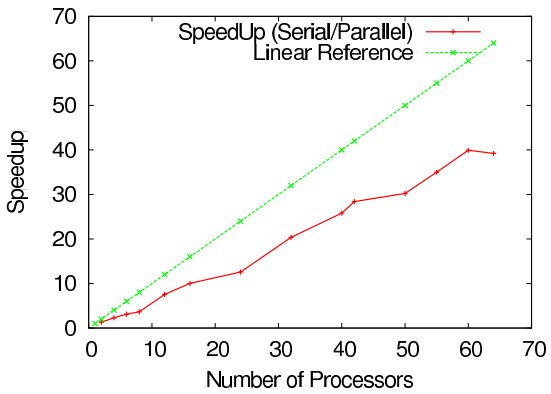
To assess the overall performance of this algorithm, some scalability results are presented in figure 11. Figure 11(a) shows that this algorithm can easily under-saturate processors, and when this happens, communication ends up dominating the overall runtime for the algorithm, which can be seen in figure 7(a), and scalability ends up being sub-linear. As the number of generators increases (as seen in figures 11(b) and 11(c)) the limit for being under-saturated is higher. Currently in the algorithm, communications are done asynchronously using non-blocking sends and receives. Also, overall communications are reduced by only communicating with a region's neighbors. This is possible because points can only move within a region radius on any two subsequent iterations, and because of this can only move into another region which is overlapping the current region. More efficiency gains could be realized through improvements in the communication, and the integration algorithms to attempt could result in linear scaling. In theory, since all of the computation is local this algorithm should scale linearly very well, up to hundreds if not thousands of processors.



(a) 40962 Generator Speedup



(b) 163842 Generator Speedup



(c) 2621442 Generator Speedup

Figure 11: Scalability results based on number of generators, Green is a linear reference where Red is the Speedup computed using parallel version of MPI-SCVT against a serial version

4. Discussion

In this paper a new algorithm is presented that makes use of existing computational geometry techniques to provide a parallel computation of a Delaunay triangulation. This new algorithm is used to create spherical centroidal Voronoi tessellations, with potential applications to climate modeling. Using a new grid generator based on this algorithm, we compare performance against a similar grid generator using a standard spherical triangulation algorithm.

A significant speedup is shown using this new algorithm for the computation of spherical Delaunay triangulations, in addition to full SCVTs. Speedups were computed against a well-known algorithm for computing spherical Delaunay triangulations [14]. Also presented is a basic method for performing fast domain decomposition on the sphere. The presented algorithm is robust enough to deal with load balancing on variable resolution meshes, and is capable (though this is not presented) of handling limited area meshes on the surface of the sphere as well. The algorithm can further be modified to work on planar grids. Also, the algorithm can potentially be modified to handle full 3D triangulations of spherical structures by some mapping techniques

The source code used for MPI-SCVT can be downloaded at:
https://sourceforge.net/projects/mpi-scvt/

5. Acknowledgements

We would like to thank Geoff Womeldorff, and Michael Duda for many useful discussions. The work of Doug Jacobsen, Max Gunzburger, John Burkardt, and Janet Peterson was supported by the US Department of Energy under grants number DE-SC0002624 and DE-FG02-07ER64432.

References

[1] P. Cignoni, C. Montani, R. Scopigno, Dewall: A fast divide and conquer delaunay triangulation algorithm in e-d, Computer-Aided Design30(1998) 333–41.

[2] A. Chernikov, N. Chrisochoides, Algorithm 872: Parallel 2d constrained delaunay mesh generation, ACM Transactions on Mathematical Software34(2008) 6:1–6:20.

[3] C. Pain, et al., Three-dimensional unstructured mesh ocean modelling, Ocean Modelling10(2005) 5–33.

[4] H. Weller, H. Weller, A. Fournier, Voronoi, delaunay, and block-structured mesh refinement for solution of the shallow-water equations on the sphere, Monthly Weather Review137(2009) 4208–24.

[5] T. Ringler, et al., Exploring a multi-resolution modeling approach within the shallow-water equations, Monthly Weather Review(2011). Accepted.

[6] S. Lloyd, Least squares quantization in pcm, IEEE Transactions on Information Theory28(1982) 129–37.

[7] Q. Du, M. Emelianenko, L. Ju, Convergence of the lloyd algorithm for computing centroidal voronoi tessellations, SIAM Journal of Numerical Analysis44(2006) 102–19.

[8] P. Bowers, W. Diets, S. Keeling, Fast algorithms for generating delaunay interpolation elements for domain decomposition, 1998. Unpublished: http://www.math.fsu.edu/ aluffi/archive/paper77.ps.gz.

[9] A. Saalfeld, Delaunay triangulations and stereographic projections, Cartography and Geographic Information Science26(1999) 289–96.

[10] J. Shewchuk, Triangle: Engineering a 2d quality mesh generator and delaunay triangulator, Applied Computational Geometry: Towards Geometric Engineering1148(1996) 203–22.

[11] Q. Du, V. Faber, M. Gunzburger, Centroidal voronoi tessellations: applications and algorithms, SIAM Rev41(1999) 637–76.

[12] N. Metropolis, S. Ulam, The monte carlo method, Journal of the American Statistical Association44(1949) 335–41.

[13] R. Heikes, D. Randall, Numerical integration of the shallow-water equations on a twisted icosahedral grid. part i: Basic design and results of tests., Monthly Weather Review123(1995) 1862–80.

[14] R. Renka, Algorithm 772: Stripack: Delaunay triangulation and voronoi diagram on the surface of a sphere, ACM Transactions on Mathematical Software23(1997) 416–34.